# University of Saskatchewan
# Department of Computer Science

## Computer Science 220
## Introduction to Digital Systems Design

## Midterm Examination
## March 2, 2004
## Closed Book, 1 Letter Size sheet of notes allowed

**Time: 75 minutes**
**Total Marks: 75**

### Instructions

1) **DO NOT OPEN THIS EXAMINATION UNTIL YOU ARE GIVEN PERMISSION!**
2) Read through the entire examination before you begin.
3) Plan your time wisely. You have 1 minute per mark.
4) This examination is closed book; one 8.5" by 11" sheet of notes is allowed.
5) Calculators, communication devices and computing devices are not permitted.
6) Answer all examination questions on this examination paper. **No other submissions will be accepted.**
7) Throughout this examination, if the identities (logical names) of inputs or outputs are specified, these identities are specified in the order *most-significant* to *least-significant*. For example, in Question 2, the inputs are specified as (A, B, C). Therefore, consider A to be the most significant input bit and C the least significant bit. **Your solutions *must* follow this convention or you will lose marks!**

MARK SUMMARY

| Question | Mark |
|---|---|
| 1 | 1,5 |
| 2 | 3 |
| 3 | 5 |
| 4 | 16 |
| 5 | 4 |
| 6 | 2 |
| 7 | 8 |
| 8 | 9 |
| 9 | 5 |
| 10 | 9 |
| Total | 55 |

73%

* Assuming positive logic (1≡1, 0≡+) throughout unless otherwise stated

1. *(2 marks)* What benefit (or benefits) do we expect to gain by using Shannon's expansion theorem?

Using Shannon's expansion facilitates the use of multiplexors and allows for lower cost implementations (ie. fewer transistors are necessary to represent the same circuit)

1.5

+. making use of an existing resource

2. *(3 marks)* Complete the following table. All values are in decimal

| N | $2^N$ |
|---|---|
| 9 | 512 |
| 11 | 2048 |
| 15 | 32768 |
| 10 | 1024 |
| 13 | 8192 |
| 7 | 128 |

3

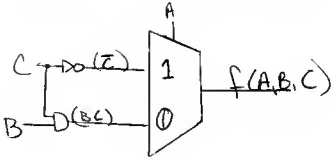
3. *(5 marks)* Implement the function f(A, B, C) = Σm(3,4,6) using a 2 input multiplexer and any other necessary gates. Show the Shannon's expansion using A as the (select) control signal to the multiplexer. **You *must* draw the circuit.**

| A \ BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |

$(A\bar{C})+(\bar{A}BC)$

$A(\bar{C})+\bar{A}(BC)$

5



4. *(10 marks)* Write the Entity and Architecture VHDL code for a logic circuit with three inputs (A, B, C) and three outputs (X, Y, Z). When the binary value of the inputs is greater than or equal to 4, the outputs are the negation of the inputs. The outputs are equal to the inputs for all other input patterns.

```
ENTITY q4·.IS
    PORT (a,b,c : IN STD_LOGIC;
            x,y,z : OUT STD_LOGIC);
END q4;

ARCHITECTURE bitassign OF q4 IS
    BEGIN.
        WITH a SELECT
        X <= (NOT(a)) WHEN a='1',
              (a) WHEN OTHERS;
        y <= (NOT(B)) WHEN a='1',
              (b) WHEN OTHERS;
        Z <= (NOT(C)) WHEN a='1',
              (C) WHEN OTHERS;

    END bitassign;
```

10

| dec | ABC | XYZ |
|---|---|---|
| 0 | 000 | |
| 1 | 001 | |
| 2 | 010 | |
| 3 | 011 | |
| 4 | 100 | 001 |
| 5 | 101 | 010 |
| 6 | 110 | 001 |
| 7 | 111 | 000 |

5. *(10 marks)* A *full-subtractor* is a combinational circuit that performs a subtraction between two bits, taking into account that a 1 may have been borrowed from a by bit of lesser significance. Design a 1-bit full-subtractor circuit to the point of developing the necessary logic equation(s). Define the inputs as **X**, **Y**, and **Bin** where **X** and **Y** are the input bits and **Bin** = **Borrow** input and define the outputs as **B** and **D** where **B** = Borrow, **D** = Difference.

**You do *not* have to draw the circuit.**

| X | Y | Bin | B | D |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

B

| X \ YBin | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 |

$B(X, Y, Bin) = (\overline{X}Y) + (YBin)$

D

| X \ YBIN | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 |

$D(X, Y, Bin) = (XBin) + (Bin) + (\overline{X}Y)$

4

not to borrow

| Bin | X | Y | B | D |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 |  | 1 | 1 |
| 1 | 1 | 1 |  |  |

$B = Bin\overline{X} + BinY + \overline{X}Y$

$D = Bin \oplus X \oplus Y$

what generates a borrow

what # plus 1 equals

X 1 1 0 0

Y 0 0 0 1

1 0 1 1

X 1 1 0 0

Y 0 0 1 1

1 0 0 1

X 1 1 1 0

Y 0 0 0 1

1 1 0 1

X 1 1 1 0

Y 0 0 1 1

1 0 1 1

6. *(8 marks)* Using full-subtractors, design a 3 bit circuit (inputs are (A, B, C)) that subtracts 3 from the input value. The input bit patterns are interpreted as an unsigned integer number. For example, if the input value is 5, the output value is 2. Underflow is permitted and ignored. In the case of underflow, the result is given by ((input - 3) mod ($2^3$)). **Draw the circuit using a box to represent each full-subtractor. Clearly identify the input and output signals.**

Input - 3

| | Input<br>A B C | Input-3<br>X Y Z | (input-3)mod($2^3$)<br>X Y Z |
|---|---|---|---|
| 0 | 0 0 0 | underflow | 1 0 0 |
| 1 | 0 0 1 | underflow | 0 1 0. |
| 2 | 0 1 0 | underflow | 0 0 1 |
| 3 | 0 1 1 | 0 0 0 | |
| 4 | 1 0 0 | 0 0 1 | |
| 5 | 1 0 1 | 0 1 0 | 1 |
| 6 | 1 1 0 | 0 1 1 | |
| 7 | 1 1 1 | 1 0 0 | |

X

| A \ BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |

$X_{(A,B,C)} = \bar{A}\,\bar{B}\,\bar{C} + ABC$

Y

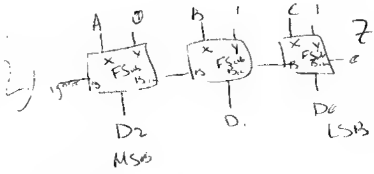
| A \ BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 |

$Y_{(A,B,C)} = \bar{B}C + AB\bar{C}$

1

Z

| A \ BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |



$Z(A,B,C) = A\bar{B}\bar{C} + B\bar{C}$

7. *(10 marks)* Implement the simplified version of the logic function f(A, B, C) = ΠM(0,1,2,5,6) using only 2 input NOR gates. **You *must* draw the circuit.**

assuming a glitch free circuit is unnecessary

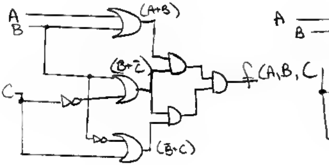
| f A\BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |

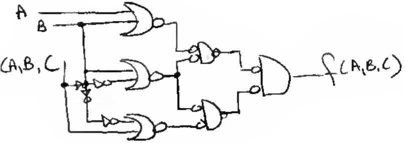
A+B

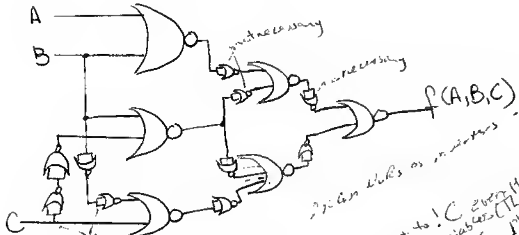
$\bar{B}+C$

$B+\bar{C}$

$$f(A,B,C)=(A+B)(B+\bar{C})(\bar{B}+C)$$

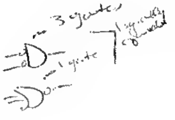






..2.

2 gates acts as inverters



$\bar{C}$ is logically equivalent to !C even though they are 2 ~~[illegible]~~ variables (Think ~~[illegible]~~ placeholders)

Will check late.

Could have assumed Inputs were available in both asserted + negated form

if input is in negated form you ... don't need to use it

8. *(10 marks)* Given the logic function f(D, C, B, A) = ΠM(3, 5, 7, 12, 13) + D(6, 11)

(a) minimize the logic function

assuming that a glitch free circuit is not required

f

| DC \ BA | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | 0 | 1 |
| 01 | 1 | 0 | 0 | D |
| 11 | 0 | 0 | 1 | 1 |
| 10 | 1 | 1 | D | 1 |

$D+\overline{C}+\overline{A}$

$D+\overline{B}+\overline{A}$

$\overline{D}+\overline{C}+\overline{B}$

~~$f(D,C,B,A)=(\overline{D}+\overline{C}+\overline{B}$~~

$f(D,C,B,A)=(\overline{D}+\overline{C}+\overline{B})(D+\overline{C}+\overline{A})(D+\overline{B}+\overline{A})$

(b) If the same function were implemented using SOP form, which cells are states that *must* be implemented by the pull-up network(s) in the circuit? Identify the states by their cell number.

cells number: 0, 1, 2, 4, 15, 14, 8, 9, 10

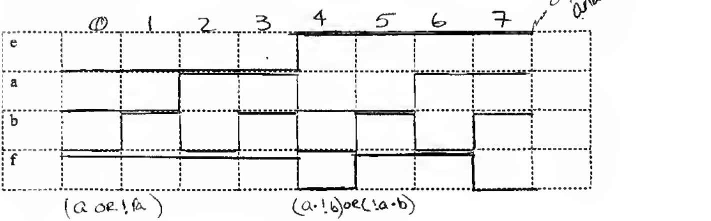
9

9. *(5 marks)* Given the following VHDL code, draw the timing diagram for the resulting circuit as if you were the Max+plus II development environment.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY midterm IS
        PORT (          a, b, e~~n~~: IN    STD_LOGIC;
                        f                   : OUT  STD_LOGIC);
END midterm;

ARCHITECTURE behavior OF midterm IS
BEGIN
        WITH e SELECT
                f <=    ((a AND !b) OR (!a AND b)) WHEN '1',
                        (a OR !a) WHEN OTHERS;
END behavior;
```



5

10. *(12 marks)* Design a circuit that accepts (as input) a 2 bit input pattern (interpreted as an unsigned binary number) and outputs a bit pattern (in 2's complement notation) that represents the (input value * (-1)). Develop your design to the point of determining the necessary logic equation(s). **You do *not* have to draw the circuit.**

assuming * is multiplication operator

Input.

| $A_1$ | $A_0$ | Output $O_1$ | $O_2$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

$O_1$

| $A_1 \backslash A_0$ | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

$O_1(A_1A_0) = A_1\overline{A_0} + \overline{A_1}A_0$

$O_2$

| $A_1 \backslash A_0$ | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 1 |

$O_2(A_1,A_0) = A_0$

if $A_1 = 0, O_1 = A_0$

if $A_1 = 1, O_1 = \overline{A_0}$

$\text{Output}_1(A_1A_0) = A_1\overline{A_0} + \overline{A_1}A_0$

$\text{Output}_2(A_1,A_0) = A_0$

Where $O_1$ = Output1 and $O_2$ = $\text{Output}_2$.

4

3 outputs.

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 |
| 2 | 1 | 1 | 0 |
| 3 | 1 | 0 | 1 |

$O_0$

| $A_1 \backslash A_0$ | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 1 |

$A_1 + A_0$.